Python Cheat Sheet

July 26, 2021

1 Basics

1.1 The print() Function:

a = 1
print('Hello world!', a)
print('Hello', end='')
print('cats', 'dogs', 'mice', sep=',')

1.2 The input() Function:

print('What is your name?')
myName = input()
print('It is good to meet you, {}'.format(myName))

1.3 The len() Function:

len('hello')

1.4 The str(), int(), and float() Functions:

str(29)
print('I am {} years old.'.format(str(29)))
int(7.7)

1.5 Boolean evaluation

if a is True:
pass
if a is not False:
pass
if a:
pass

1.6 for Loops and the range() Function

The range() function can also be called with three arguments. The first two arguments will be the start and stop values, and the third will be the step argument. The step is the amount that the variable is increased by after each iteration.
for i in range(0, 10, 2): print(i)

2 Exception Handling

def spam(divideBy):
try:
return 42 / divideBy
except ZeroDivisionError as e:
print('Error: Invalid argument: '.format(e))

2.1 Final code in exception handling

Code inside the finally section is always executed, no matter if an exception has been raised or not, and even if an exception is not caught.
def spam(divideBy):
try:
return 42 / divideBy
except ZeroDivisionError as e:
print('Error: Invalid argument: '.format(e))
finally:
print("– division finished –")

3 Lists

spam = ['cat', 'bat', 'rat', 'elephant']
spam[0] //cat
spam[-1] //elephant
'The {} is afraid of the .'.format(spam[-1], spam[-3])
len(spam)
del spam[2]

3.1 Getting Sublists with Slices:

slicing the first and second parameter out of the list **spam[1:3] //bat,rat**
spam[0:-1] //cat,bat,rat

3.2 List Concatenation and List Replication

[1, 2, 3] + ['A', 'B', 'C']
X', 'Y', 'Z'
***3**

3.3 Looping Through Multiple Lists with zip()

name = ['Pete', 'John', 'Elizabeth']
age = [6, 23, 44]
for n, a in zip(name, age):
print('{} is {} years old'.format(n, a))

3.4 The in and not in Operators

'cat' in spam
'cat' not in spam

3.5 The Multiple Assignment Trick

cat = ['fat', 'orange', 'loud']
size, color, disposition = cat

3.6 List Methods

- **append():**
  spam.append('abv') //Add it after the last item in list
- **insert():**
  spam.insert(1, 'chicken') //specify where to add it
- **index(''):**
  spam.index('cat') : search
- **remove():**
  If the value appears multiple times in the list, only the first instance of the value will be removed.
  spam.remove('cat') : remove item from list
- **sort():**
  spam.sort()
  spam.sort(reverse=True)
  spam.sort(key=str.lower)
  You can use the built-in function sorted to return a new list:
  sorted(spam)

4 Tuples

The main way that tuples are different from lists is that tuples, like strings, are immutable.
eggs = ('hello', 42, 0.5)
Convert list to tuple:
tuple(['cat', 'dog', 5])
tuple to list:
list(('cat', 'dog', 5))
list('hello') //h,e,l,l,o

5 Dictionaries

cat = 'size': 'fat', 'color': 'gray', 'disposition': 'loud'

5.1 Dictionary Methods

- keys():
  for k in cat.keys():
  print(k) //size color disposition

- values():

  for v in cat.values() :
  print(v) // fat gray loud

- items():

  for i in cat.items():
  print(i)//('size': 'fat'), ('color': 'gray'), ('disposition': 'loud')

- get():
  Get has two parameters: key and default value if the key did not exist.
  'I am bringing cups.'.format(str(cat.get('cups', 0)))

- setdefault():
  spam.setdefault('eyes', 'black')

5.2 Merge two dictionaries

x = 'a': 1, 'b': 2
y = 'b': 3, 'c': 4
z = **x, **y
z
'c': 4, 'a': 1, 'b': 3

6 Sets

A set is an unordered collection with no duplicate elements. Basic uses include membership testing and eliminating duplicate entries. Set objects also support mathematical operations like union, intersection, difference, and symmetric difference.
When creating an empty set, be sure to not use the curly braces or you will get an empty dictionary instead.
And as an unordered data type, they can't be indexed.
s = 1, 2, 3
s = set([1, 2, 3])

6.1 Sets Methods

- add(): **s.add(4)**
  Using the add() method we can add a single element to the set

- update():
  s.update([2, 3, 4, 5, 6])
  multiple ones and automatically remove duplicates

- remove()/discard():
  s.remove(3)
  s.discard(3)
  Both methods will remove an element from the set, but remove() will raise a key

error if the value doesn't exist.

- intersection():
  s1 = 1, 2, 3
  s2 = 2, 3, 4
  s3 = 3, 4, 5
  s1.intersection(s2, s3)
  will return a set containing only the elements that are common to all of them.

- union():
  s1 = 1, 2, 3
  s2 = 3, 4, 5
  s1.union(s2)
  will create a new set that contains all the elements from the sets provided.

- difference():
  will return only the elements that are unique to the first set
  s1.difference(s2)

- symetric_difference():
  s**1.symmetric_difference(s2)**
  will return all the elements that are not common between them.

7 itertools Module